Звіт

з лабораторної роботи №7

з курсу алгоритми і методи обчислень.

(АЛГОРИТМИ ПОШУКУ В ЛІНІЙНИХ СТРУКТУРАХ)

Студента 1-го курсу,

Парфенюка Тимофія Павловича

2019

Завдання:

забезпечити пошук вказаних елементів у невідсортованих масивах. Здійснити їх сортування. Виконати пошук у відсортованих масивах. Оцінити кількість порівнянь елементів в обох випадках.

Індивідуальне завдання: елементи масиву А в одному екземплярі, котрі присутні в масиві B;

**Код:**

using System;

using System.Collections.Generic;

namespace Search

{

class Program

{

static int p = 0;

static public void Sort(int[] mass)

{

int i, j, step, t1 = 1, t2 = 0, tmp;

t1 = t1 + 1;

for (step = mass.Length / 2; step > 0; step /= 2, t1++)

{

for (i = step; i < mass.Length; i++, t1++)

{

tmp = mass[i];

t1 = t1 + 1;

for (j = i; j >= step; j -= step, t1++)

{

if (tmp < mass[j - step])

{

mass[j] = mass[j - step];

t2 = t2 + 1;

t1 = t1 + 1;

}

else

break;

}

mass[j] = tmp;

t1 = t1 + 1;

}

}

}

static public int InputInt(string s)

{

int input = 0;

string cont = "";

do

{

try

{

cont = "";

Console.WriteLine(s);

input = int.Parse(Console.ReadLine());

}

catch (Exception e)

{

Console.WriteLine(e.Message + "Again?");

cont = Console.ReadLine();

}

} while (cont == "yes");

return input;

}

static public int[] CreateMas(int dim)

{

int[] Mas = new int[dim];

for (int i = 0; i < Mas.Length; i++)

{

Mas[i] = InputInt("Input " + i + "-th element");

}

return Mas;

}

static public void PrintMas(int[] Mas)

{

for (int i = 0; i < Mas.Length; i++)

{

Console.Write($"{Mas[i]} ");

}

Console.WriteLine("");

}

static public int LinealSearch(int[] arr1, int key)

{

int[] arr = new int[arr1.Length];

Array.Copy(arr1, arr, arr1.Length);

Array.Resize(ref arr, arr.Length + 1);

arr[arr.Length - 1] = key;

int i = 0;

while (arr[i] != key)

{

p++;

i++;

}

return i;

}

static public bool SortedTest(int[] arr)

{

bool sorted = true;

for (int i = 0; i < arr.Length - 1; i++)

{

if (arr[i] < arr[i + 1])

sorted = true;

else return false;

}

return sorted;

}

public static int BinarySearch(int[] arr, int key)

{

if (SortedTest(arr) == false)

{

Sort(arr);

}

int L = 0;

int R = arr.Length - 1;

while (L <= R)

{

int m = (L + R) / 2;

if (key == arr[m])

{

p++;

return m;

}

else if (key < arr[m])

{

p++;

R = m - 1;

}

else

{

p++;

L = m + 1;

}

}

return -1;

}

static public bool Repeat(int[] arr, int key)

{

int i = 0;

int c = 0;

while (i < arr.Length)

{

if (arr[i] == key)

{

c++;

}

i++;

if (c >= 2) return false;

}

return true;

}

public static void Find(int[] arr1, int[] arr2)

{

List<int> ind = new List<int>();

for (int i = 0; i < arr1.Length; i++)

{

int index = LinealSearch(arr2, arr1[i]);

if (index == arr2.Length) { }

if (index < arr2.Length)

{

if (Repeat(arr1, arr2[index]))

{

index = LinealSearch(arr1, arr2[index]);

bool add = true;

for (int j = 0; j < ind.Count; j++)

{

if (ind[j] == index)

{

add = false;

break;

}

else add = true;

}

if (add)

{

ind.Add(index);

//Console.WriteLine(index);

}

}

}

}

if (ind.Count != 0)

{

Console.WriteLine("Indexes of elements in array");

foreach (int i in ind)

{

Console.Write(i + " ");

}

}

else Console.WriteLine("Elements is not found");

}

static void Main(string[] args)

{

try

{

int[] arr1 = CreateMas(InputInt("Input number of nubers"));

int[] arr2 = CreateMas(InputInt("Input number of numbers"));

Find(arr1, arr2);

Console.WriteLine();

Console.WriteLine("------------------------------------------------");

Console.WriteLine($"Numbers of compare: {p}");

PrintMas(arr1);

PrintMas(arr2);

}

catch (Exception e)

{

Console.WriteLine($"{e} + /n Something wrong!");

}

}

}

}

***Висновок:***

*Лінійний пошук*

***Ідея***

Проглядати почергово елементи масиву, доки не буде знайдено шуканий елемент або не буде досягнуто кінець масиву.

Це самий простий вид пошуку деякого елемента серед інших, що здійснюється за допомогою перевірки кожного елемента до тих пір, поки вони не будуть збігатися. Загальна ідея цього виду пошуку така: усі елементи розглядаються послідовно, один за одним. Це дає змогу не пропустити жодного елемента. Якщо збіг буде знайдено, то пошук припиняється і його результат є позитивним. Якщо не знайдено, то результат буде негативним.

Перевагами такого пошуку є простота його реалізації, він не потребує додаткового об’єму пам’яті або додаткової роботи з функціями. Це дозволяє працювати вже під час отримання даних.

Також існує певний покращений послідовний алгоритм, який пришвидшує пошук. У множині встановлюється бар’єр, тобто елемент, який задовольняє пошуку. У циклі відпадає необхідність перевірки умови, зв’язаної з границями множини. Таким чином буде обмежена зміна індексу.

***Ідея алгоритму бінарного пошуку така:***

• порівняти аргумент пошуку ключ зі значенням середнього елемента х [середовищ]масиву х,

де середовищ = [П / 2], а [з] ціла частина числа с

• якщо вони рівні, то пошук завершено, інакше, якщо ключ < х [середовищ], виконати аналогічним чином пошук в позиціях масиву, попередніх позиції середовищ, в іншому випадку, якщо ключ> х [середовищ], виконати аналогічним чином пошук в позиціях масиву х, Наступних за позицією середовищ

Виключити з подальшого розгляду частина масиву дозволяє той факт, що масив впорядкований

Кожне порівняння зменшує число можливих кандидатів у 2 рази Максимальне число кроків пошуку буде в тому випадку, коли аргумент пошуку знаходиться на початку або в кінці масиву У цьому випадку буде потрібно log2n + 1 ітерацій Дійсно, якщо число елементів у масиві дорівнює п = 2т, ключ буде знайдений, коли нерозглянутим залишиться тільки один елемент, тобто через т кроків У свою чергу, при заданому п маємо т = log2n Після аналізу останнього елемента отримуємо загальне число ітерацій log2n + 1 Тому обчислювальна складність бінарного пошуку складає O (log2n)

*Однак наведений алгоритм не дозволяє в загальному випадку точно вирішити завдання пошуку, коли файл або масив містять повторювані значення ключів*

***Послідовний пошук не гарантує знаходження елементу. Також і бінарний. Тільки лінійний пошук з барєром гарантує знаходження елементу,.***

**Лінійний пошук:**

**Первага:** лінійний пошук не вимагає додаткової пам'яті або обробки/аналізу функції, так що може працювати в потоковому режимі при безпосередньому отриманні даних з будь-якого джерела. Так само, лінійний пошук часто використовується у вигляді лінійних алгоритмів пошуку максимуму/мінімуму.

**Недолік:** У зв'язку з малою ефективністю в порівнянні з іншими алгоритмами лінійний пошук зазвичай використовують лише тоді, коли відрізок пошукової системи містить дуже мало елементів

**Бінарний пошук:**

**Переваги:** Суттєво швидший за лінійний пошук

**Недоліки**: Працює тільки на впорядкованих множинах.

***Вхідні та вихідні данні:***

